Bokningskalender

Andreas Grönlund

Examensarbete för ingenjör (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2016

EXAMENSARBETE

Författare:	Andreas Grönlund
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Kaj Wikman

Titel: *Bokningskalender*

Datum 23.3.2016 Sidantal 37

Abstrakt

Detta examensarbete gjordes för Österbottens museum som behövde en bokningskalender med svenska, finska och engelska som språkval till deras webbsida. Den skulle ge användare möjlighet att från Österbottens museum, Kuntsi museum för modern konst eller Tikanojas konsthem välja utställning och boka en guidad rundtur till denna.

Kalenderns dynamiska delar gjordes med JavaScript samt PHP kombinerat med SQL och en MySQL databas. Delar av arbetet använder HTML5 elementet canvas för grafiska objekt.

Resultatet blev en responsiv webbapplikation där användare kan göra bokningar och ett admingränssnitt för att administrera utställningar. Via admingränssnitt kommer användaren även åt en bokningslista och statistik över bl.a. antalet bokningar.

Språk: svenska Nyckelord: HTML, HTML5, PHP, SQL, JavaScript, kalender

---

OPINNÄYTETYÖ

Tekijä: Andreas Grönlund
Koulutusohjelma ja paikkakunta: Tietotekniikka, Vaasa
Ohjaajat: Kaj Wikman

Nimike: Varauskalenteri

Päivämäärä 23.3.2016 Sivumäärä 37

Tiivistelmä

Tämä opinnäytetyö on tehty Pohjanmaan museolle joka tarvitsi verkkosivuilleen varauskalenterin ruotsiksi, suomeksi ja englanniksi. Kalenterin täytyi suoda käyttäjälle mahdollisuus valita näyttely ja varata opastettu kierros Pohjanmaan museossa, Kuntsin modernin taiteen museossa tai Tikanojan taidekodissa.

Kalenterin dynaamiset osat tehtiin JavaScriptillä ja PHP:llä yhdistettynä SQL ja MySQL tietokantaan. Osa työstä käyttää HTML5 canvas elementtiä graafisille objekteille.

Lopputulos on sulavasti toimiva verkkosovellus jossa käyttäjä voi tehdä varauksia sekä järjestelmänhallintatyökalu näyttelyiden hallinnointiin. Järjestelmänhallintatyökalun kautta käyttäjä voi myös tulostaa varauskirjan sekä tilastoja mm. varausten määrästä.

Kieli: ruotsi Avainsanat: HTML, HTML5, PHP, SQL, JavaScript, kalenteri

---

BACHELOR'S THESIS

Author: Andreas Grönlund
Degree Programme: Information Technology, Vaasa
Supervisor: Kaj Wikman

Title: Booking calendar

Date 23.3.2016 Number of pages 37

Summary

This thesis was made for the Ostrobothnian museum that needed a booking calendar with swedish, finnish and english as language choices for their website. The requirements were that it would give users the ability to choose between the Ostrobothnian museum, Kuntsi Museum of Modern Art and the Tikanoja Art Museum and book a guided tour.

The calendars dynamic parts were made with JavaScript and PHP combined with SQL and a MySQL database. Portions of the web application uses the HTML5 canvas element for drawing graphical objects.

The result is a responsive web application where users can make bookings and an admin interface for managing exhibitions. From the admin interface, the user can also access a reservation list and view statistics such as the number of bookings.

Language: swedish Key words: HTML, HTML5, PHP, SQL, JavaScript, calendar

---

Innehållsförteckning

Ordförklaringar

JQuery	Ett JavaScript bibliotek med funktioner för att modifiera innehållet på webbsidor.
HTML	Ett märkspråk för att skapa och strukturera webbsidor.
HTML5	Utökar HTML-standarden med nya element och attribut.
SQL	Programmeringsspråk för att hämta och modifiera information i databaser.
MySQL	Databashanterare som använder programmeringsspråket SQL.
Cookies	Datafil som sparas av webbläsare och innehåller exempelvis språkpreferenser hos en besökare.
JavaScript	Programmeringsspråk som körs på klientsidan och används exempelvis för att öka funktionalitet på webbsidor.
Chart.js	HTML5 och *canvas* baserat diagram- och visualiseringsverktyg.
YUI	JavaScript och CSS-bibliotek för att skapa interaktiva webbapplikationer.
Canvas	HTML5-element för att rita grafiska objekt.

1 Inledning

Målet för examensarbetet var att skapa en bokningskalender i form av en webbapplikation till Österbottens museums webbsida. Applikationen skulle ge företag och besöksgrupper möjlighet att boka guidade rundturer till utställningar vid Kuntsi museum för modern konst, Tikanojas konsthem och Österbottens museum. Webbapplikationen skulle även ge möjlighet att lägga till nya och uppdatera existerande utställningar, administrera samt visa statistik över tidigare bokningar.

1.1 Uppdragsgivare

Österbottens museum upprätthålls av Vasa stad och fungerar som Vasa stads historiska museum, landskapsmuseum, regionkonstmuseum och naturvetenskapligt museum. Museet fullföljer sitt uppdrag i enlighet med museilagen och är underordnat Undervisnings- och kulturministeriet. Österbottens museum är en del av den riksomfattande museiverksamheten i Finland, som koordineras och styrs av Museiverket.

Österbottens museum verkar inom områdena kulturhistoria, konst och naturvetenskaper. Varje område sköts av en egen avdelning. Museet leds av en museidirektör och avdelningarna leds av intendenter. I museet arbetar dessutom amanuenser, forskare och konservatorer inom olika områden.

Museet hör till Vasa stads bildningssektor och lyder under bildningsdirektören. För museets förvaltning ansvarar museinämnden, som utnämns av stadsfullmäktige.

Österbottens museum grundades år 1895 och upprätthölls till en början av Föreningen för Österbottens historiska museum r.f. . Vasa stad har upprätthållit museet sedan år 1990.

1.2 Bakgrund

iMG Media har vid flera tillfällen kontaktats angående en kalenderapplikation. Då Österbottens museum meddelade sitt intresse för en bokningskalender till deras webbsida beslöts att en sådan skulle göras. En lista med de specifikationer och funktioner som applikationen skulle ha utarbetades. I samband med att iMG Media skapade museets webbsida hade man redan tidigare kommit överens om en liknande applikation, men på grund av tidsbrist förverkligades den aldrig.

1.3 Uppgift

Uppgiften blev att göra en responsiv webbapplikation där användare från en kalender skulle kunna boka guidade rundturer. Det skulle även finnas ett admingränssnitt med följande funktioner:

- Lägga till samt editera existerande utställningar för de fyra museerna.
- Telefonbokningar där museets personal kan göra en bokning.
- En filtrerbar lista över alla bokningar.
- Möjlighet att radera bokningar.
- Statistik över gjorda bokningar
- Applikationen skulle även finnas på svenska, finska och engelska.

Som botten användes en tidigare utvecklad bokningsapplikation som vidareutvecklades med de nya funktionerna samt dagens standard för funktionalitet och layout.

2 Tekniker

Webbapplikationen använder flera olika tekniker och programmeringsspråk för den funktionalitet som skulle finnas. Dessa är på klientsidan HTML, HTML5, CSS samt JavaScript och på serversidan PHP samt MySQL.

2.1 HTML

HTML står för *HyperText Markup Language* och är det märkspråk som webbsidor i allmänhet skrivs på. En typisk grundstruktur för en HTML-sida ser ut på följande sätt.

Kodexempel 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Sidans titel</title>
</head>
<body>
  <h1>Sidans rubrik</h1>
  <p>En paragraf på sidan</p>
</body>
</html>
```

Figur 1 HTML-sida

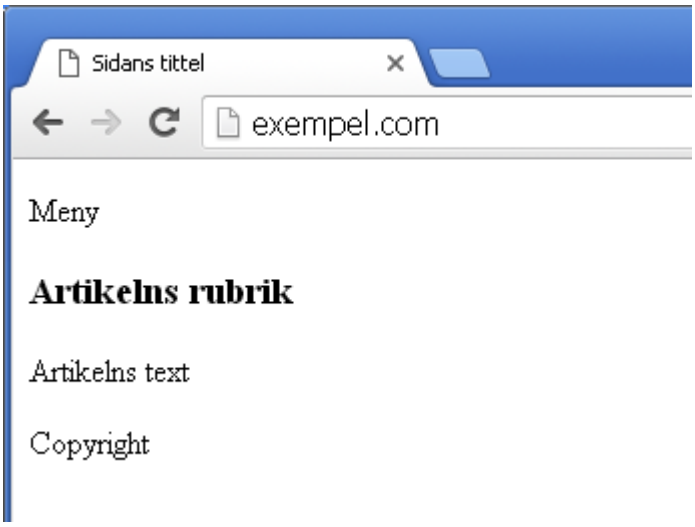
Den första raden anger dokumenttypen som HTML enligt 4.01 standarden och följande rad anger att valideringen sker enligt *strict dtd, document type declaration,* standarden. En annan vanlig *dtd* är *Transitional* som även innehåller stöd för alla de HTML-element som finns i *strict* samt de som fasats ut från den. HTML-taggen anger vilken del av sidan som innehåller HTML-element. *Head* och *body* avgränsar innehållet där *head* innehåller exempelvis information som sidans rubrik, metadata samt länkar till filer som inkluderas till sidan. *Title* är HTML-taggen som innehåller sidans rubrik. Rubriken visas bland annat som namn på en flik i en webbläsare. *Body* innehåller själva innehållet på sidan. Dessa placeras i HTML-element som *h1* och *p*, där *h1* innehåller rubriker och *p* innehåller paragrafer. (*Conformance: requirements and recommendations,* 2016)

2.2 HTML5

HTML5 utökade den tidigare HTML-standarden och introducerade bl.a. nya HTML-element som delar av examensarbetet använder.

Kodexempel 2

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sidans titel</title>
  </head>
  <body>
<nav><p>Meny</p></nav>
   <section>
    <article>
     <header><h1>Artikelns rubrik</h1></header>
     <p>Artikelns text</p>
    </article>
   </section>
  </body>
 <footer>Copyright</footer>
</html>
```



Figur 2 HTML5-sida

I HTML5 har dokumenttypen och *dtd* ersatts av *html* som säger att sidan följer HTML5-standarden. Dessa webbsidor har vanligtvis även *meta* taggen *charset* som säger vilken teckenuppsättning sidan använder. Exempel på de nya elementen som introducerats är *header* för rubriker och *nav* för exempelvis menyer. Med det nya *section* elementet kan man vidare dela in en sida med exempelvis *article*. Artiklar innehåller då en egen *header* för rubrik och *p* för text. Copyright information som vanligtvis finns längst ner på sidan kan placeras i HTML5-elementet *footer*. Fördelen med de nya elementen är bl.a. att webbläsare har enklare att förstå innehållet på en sida och koden blir mer lättläst för utvecklare. (*HTML5 Introduction*, 2016)

2.3 CSS

CSS står för *Cascading Style Sheets* och används för att beskriva presentationsstilen för webbsidor. Exempelvis typsnitt, fontstorlek och färger sätts med hjälp av stilmallar.

Kodexempel 3

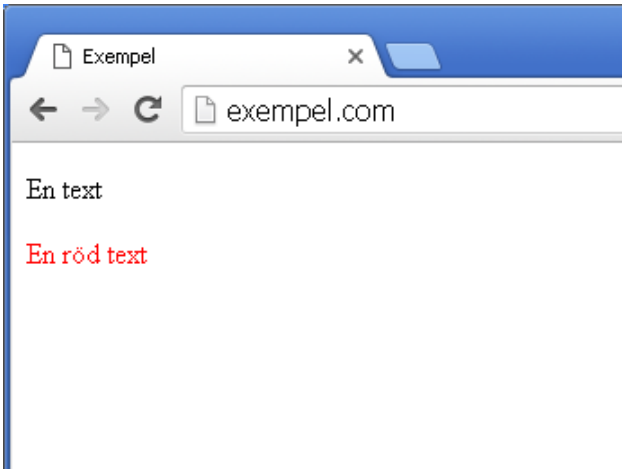
```
<p>En text</p>
<p class="röd">En röd text</p>
```

I exemplet ovan kan man ändra färgen på den andra paragrafen med följande CCS kod.

Kodexempel 4

```
.röd {
	color: red;
}
```

Detta ger följande resultat i en webbläsare:



Figur 3 CSS-exempel

Genom att samla all presentationsinformation i CSS-filer går det därmed enkelt att göra förändringar på alla de sidor som inkluderar CSS-filen genom att endast ändra innehållet i en fil. (*CSS Introduction*, 2016)

2.4 PHP

PHP, rekursiv akronym för *PHP: Hypertext Preprocessor* är ett populärt skriptspråk som körs på webbservrar för att få innehållet på webbsidor dynamiskt. För att köra PHP-kod på en sida vars webbserver stöder PHP behöver man endast ha ändelsen .php på filen och placera koden innanför start taggen *<?php* och slut taggen *?>*. Föregående exempel gjord med PHP skulle se ut som följande. (Beighley & Morrison, 2014, s. 30)

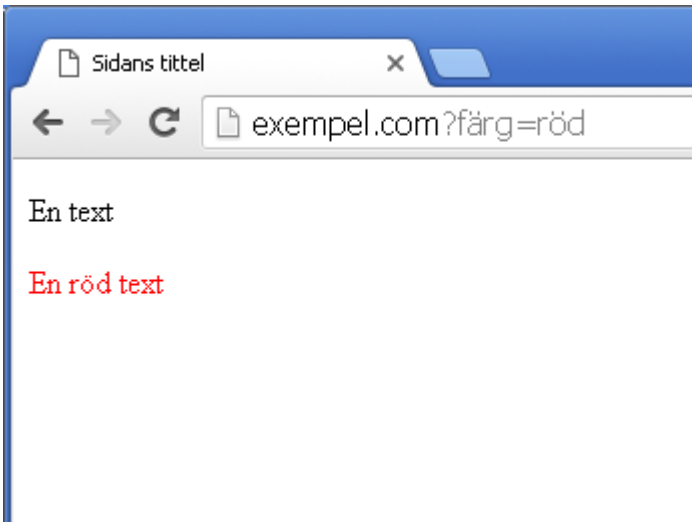
Kodexempel 5

```
<p> <?php echo "En text"; ?></p>
<p class="röd"> <?php echo "En röd text"; ?></p>
```

För att få exemplet dynamiskt kan man använda PHP-funktionen *$_GET*. Funktionen läser innehållet i adressfältet som sedan kan användas direkt med funktionen eller som i följande exempel sparas i en egen variabel. (Beighley & Morrison, 2014, s. 276)

Kodexempel 6

```
$textExempel = "En text" ;
$färgExempel = $_GET['färg'];

<p> <?php echo $textExempel; ?></p>
<p class="röd">En <?php echo $färgExempel; ?> text</p>
```



Figur 4 PHP-exempel

Den första textsträngen sparas i variabeln *$textExempel* som skrivs ut med funktionen *echo*. För att visa texten på nästa rad skrivs "En" och "text" ut som vanlig text men "röd" hämtas från webbläsarens adressfält med funktionen *$_GET*. Funktionen läser värdet för färg, som i webbläsaren är röd, och sparar denna i variabeln *$färgExempel*. Denna skrivs sedan ut med *echo* funktionen.

2.5 SQL

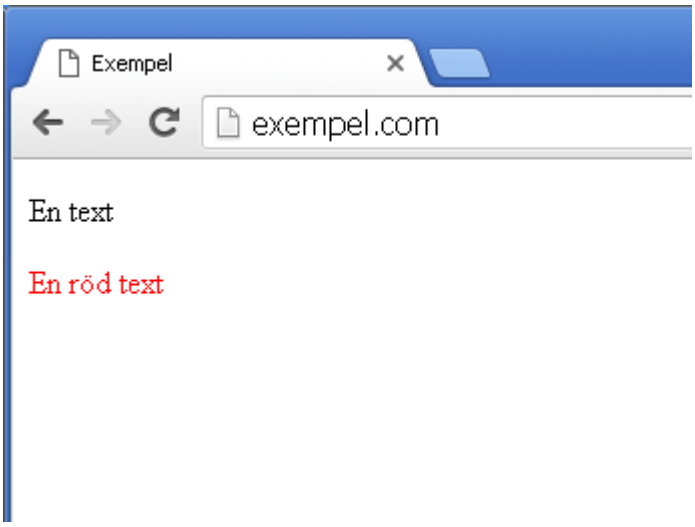
SQL, som står för *Structured Query Language*, är ett standardiserat programspråk för att hämta och modifiera data i en relationsdatabas. De föregående exemplen kunde vidare göras mer dynamisk ifall texten eller delar av den hämtades från en databas. Databasen kunde exempelvis ha en färgtabell där en användare kunde spara olika färger. Dessa kunde sedan hämtas med SQL för att sedan skrivas ut på sidan. En modifierad version av föregående exempel som använder SQL och PHP-funktionen *mysql_query()* istället för *$_GET* skulle se ut som följande. (Beighley & Morrison, 2014, s. 88)

Tabell 1 Exempel på en tabell med färger.

id	färg
1	röd
2	grön
3	blå
…	…

Kodexempel 7

```
$textExempel = "En text" ;

$rFärg = mysql_query("SELECT färg FROM tblFärger WHERE id = 1");
$färgExempel = mysql_fetch_array($rFärg);

<p> <?php echo $textExempel; ?></p>
<p class="röd">En <?php echo $färgExempel['färg']; ?> text</p>
```



Figur 5 SQL-exempel

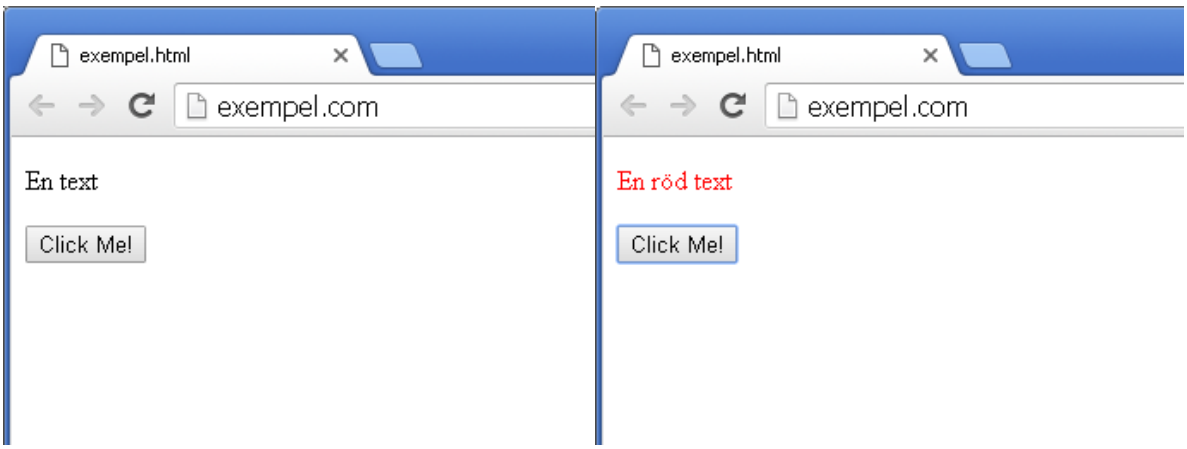
Den första raden skrivs ut som i föregående exempel men ordet "röd" i den andra raden hämtas nu från tabellen tblFärger. Detta görs med PHP-funktionen *mysql_querry()* som kör SQL-satsen *SELECT*. I tabellen finns det två kolumner, en för id och en för färger. Satsen säger att från tabellen tblFärger skall kolumnen färg hämtas och från den skall den rad vars värde för id är 1 hämtas. För att hantera den information som *mysql_query()* returnerar används PHP-funktionen *mysql_fetch_array()*. Den lagrar informationen från *$rFärg* som en räcka i den nya variabeln *$färgExempel* där en räcka kan beskrivas som en lista med data. Från denna lista hämtas den information som skrivs ut i webbläsaren med *echo*. I motsats till att skriva ut en variabel som tidigare så är syntaxen annorlunda då variabeln är en räcka. För att komma åt den information vi vill ha, texten "röd", så specificeras positionen i *$färgExempel* listan med klammrar. Formatet blir då *$variable['position']* dvs. *$färgExempel['färg']*. Alternativt går det även att skriva *$färgExempel[0]*. Båda returnerar samma information men det första alternativet är mer lättläst. Denna variabel går nu som tidigare att skrivas ut med *echo*. (Beighley & Morrison, 2014, s. 135)

2.6 JavaScript

JavaScript är ett prototyp-baserat skriptspråk och används främst på klientsidan i webbtillämpningar, det vill säga utförs i en webbläsares JavaScript-motor. En variant av föregående exempel går att göra dynamisk med JavaScript-funktionen *onclick()*. (Haverbeke, 2014, s. 224) samt en funktion kallad *myFunktion()*. (Haverbeke, 2014, ss. 41, 233)

Kodexempel 8

```
<!DOCTYPE html>
<html>
<body>
<p id="röd">En text</p>

<button type="button" onclick="document.getElementById('demo').innerHTML = 'En röd
text';myFunction()">Click Me!</button>

<script>
 function myFunction() {
    var x = document.getElementById("röd");
    x.style.color = "red";
}
</script>

</body>
</html>
```



Figur 6 Före och efter aktivering av Click me!.

Strukturen är standard HTML5 med elementen *p* och *button*. Paragraf elementet *p* har taggen *id* som identifierar den med namnet "röd". *Id* i motsats till *class* som användes i de tidigare

exemplen ger ett unikt namn och kan därför endast användas för att identifiera ett specifikt element. Flera element kan däremot ha samma *class*.

3 Bokningskalender

Bokningskalendern är uppdelad i tre olika webbsidor. Den första sidan innehåller själva webbapplikationen med kalendern, filtreringsalternativen och ett formulär för kontaktuppgifter. På den andra sidan visas statistik över bokningar och på den tredje en lista över aktuella bokningar. Applikationen har även en administratorgränsnitt som användaren kommer åt genom att logga in på sidan. Som administratör har användaren tillgång till funktioner som att lägga till nya och editera existerande utställningar samt en kontrollpanel.

3.1 Syfte

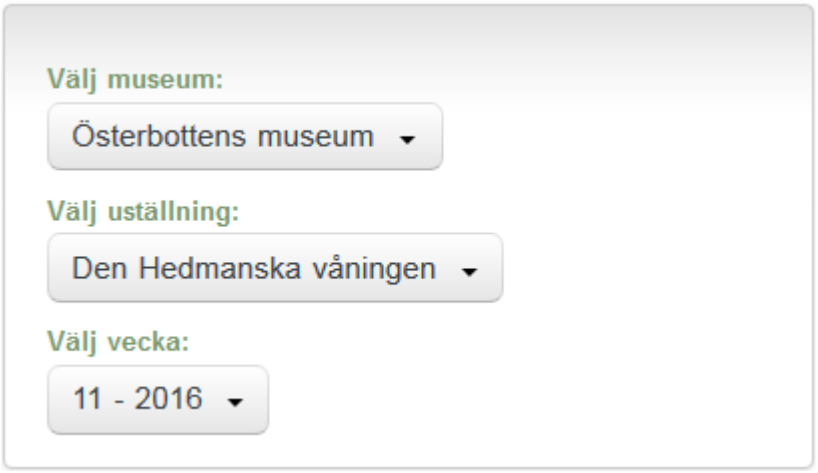
Syftet med webbapplikationen är att ge möjlighet att från en filtrerbar kalender boka en eller flera tider till guidade rundturer. På Österbottens museum sköter samma person alla guidningar, vilket betyder att då en tid för en viss utställning blir bokad visas samma tid för museets andra utställningar som bokade.

3.2 Funktioner

Webbapplikationen är uppdelad i tre delar där den första innehåller filtreringsalternativ i form av tre rullgardinsmenyer från vilka användaren väljer museum, utställning samt vecka. Den andra delen är själva kalendern som visar de bokade samt de bokningsbara tiderna av vilka användaren kan markera en eller flera tider. Den tredje innehåller formuläret för kontaktuppgifter.

3.2.1 Filtreringsalternativ

Kalendern visar de bokningsbara tiderna då användaren har valt museum, utställning och vecka från de tre rullgardinsmenyerna.



Figur 7 rullgardinsalternativen

Den första menyn listar alla museer. Listan med museer hämtas från en statisk PHP-räcka. Den andra menyn, som aktiveras först då den första fått ett värde, innehåller en lista med utställningar vid det valda museet. Listan är dynamisk och innehållet hämtas från tabellen tblExhebitions i databasen.

Tabell 2 tblExhebitions

id	descriptionse	descriptionfi	descriptioneng
1	Den Hedmanska våningen	Hedmanin kerros	The Hedman floor
2	Vasa 400	Vaasa 400	Vaasa 400
3	Växlande utställning	Vaihtuva näyttely	Temporary exhibition
7	Terranova	Terranova	Terranova
8	UN/SAFE	UN/SAFE	UN/SAFE
9	Före kriget - lakttagelser om tiden	Före kriget - lakttagelser om tiden	Före kriget - lakttagelser om tiden

Tabellen tblExhebitions innehåller alla de utställningar som finns och är länkade till ett museum via kolumnen museumId. Listan går att editera via webbsidans admingränssnitt.

3.2.2 Meny vecka

Den tredje rullgardinsmenyn, som också aktiveras först då användaren valt ett museum, innehåller en lista med veckor. Listan visar 30 veckor framåt räknat från nuvarande vecka som väljs automatiskt.

Österbottens museum ville ha en kalender som är indelad enligt vecka. Problemet som uppstår med en sådan kalender är att ett år inte har 52 veckor eftersom en vecka tekniskt sett är 7.14 dagar lång. För att kompensera kan man ha 53 veckor på ett år men problemet som då uppstår är att t.ex. år 2015 slutade 31.12, som var en torsdag, och definitionen på årets första vecka är:

"If 1 January is on a Monday, Tuesday, Wednesday or Thursday, it is in week 01. If 1 January is on a Friday, it is part of week 53 of the previous year; if on a Saturday, it is part of week 52 (or 53 if the previous year was a leap year); if on a Sunday, it is part of week 52 of the previous year." (ISO week date, 2015)

Det skulle gå att lösa problemet genom att helt enkelt öka kalendern till 53 veckor, förutom att vecka 53 inte är en hel vecka utan 31.12, som var en torsdag, var den sista dagen och följande dag var 1.1.2016. Detta skulle då betyda att ifall någon hade bokat en tid 30.12.2015 skulle de ha gjort det i vecka 53 på ett år med 52 hela veckor, men ifall någon däremot hade valt att boka en tid mellan 1.1 och 3.1.2016 skulle bokningen ha varit för vecka 53 år 2015 alternativt vecka 0, år 2016, då vecka 1 officiellt började först 4.1.2016.

Ett annat problem som uppstår med en veckovis kalender är antalet veckor. Ifall man har en lista med 52 veckor, och vill boka ett datum i början av 2016 måste listan ökas på för att inkludera dessa datum. Det skulle betyda att istället för att år 2015 slutar vid vecka 52 fortsätter listan med vecka 53 - 2016 och vecka 1 - 2016, alternativ vecka 53 - 2015, vecka 0 - 2016 och vecka 1 - 2016.

Ingen av alternativen skulle underlätta för användaren att avgöra vilket datum som hör till vilken vecka. Lösningen för kalendern blev att öka listan till 53 veckor och definiera de resterande dagarna för 2015 som vecka 53 och de första dagarna i januari 2016 som vecka 0.

3.2.3 Språkval

Applikationen finns på tre språk, svenska, finska och engelska. All text som inte hämtas från databasen finns lagrad i en av de tre språkfilerna lang-se.php, lang-fi.php och lang-eng.php. Då en besökare öppnar webbsidan körs innehållet i filen common.php som kontrollerar ifall det finns en session cookie. Ifall inte så skapas en med information om bl.a. det valda språket och om inget språk valts sätts finska som standard. Beroende på innehållet i session cookie filen laddar common.php sedan en av språkfilerna och sidan visas på det språket.

Det fjärde värdet i t.ex. räckan lang-se.php innehåller texten "Välj vecka" och är sparad i formatet *$langarray[3] = 'Välj vecka';*. På engelska *$langarray[3] = 'Choose week';*. Orsaken till att det står [3] är för att en räcka startar från värdet 0. Textraden visas sedan på sidan med PHP-funktionen *echo*.

Kodexempel 9

```
<li><p><?php echo $langarray[3] ?>:</p></li>.
```

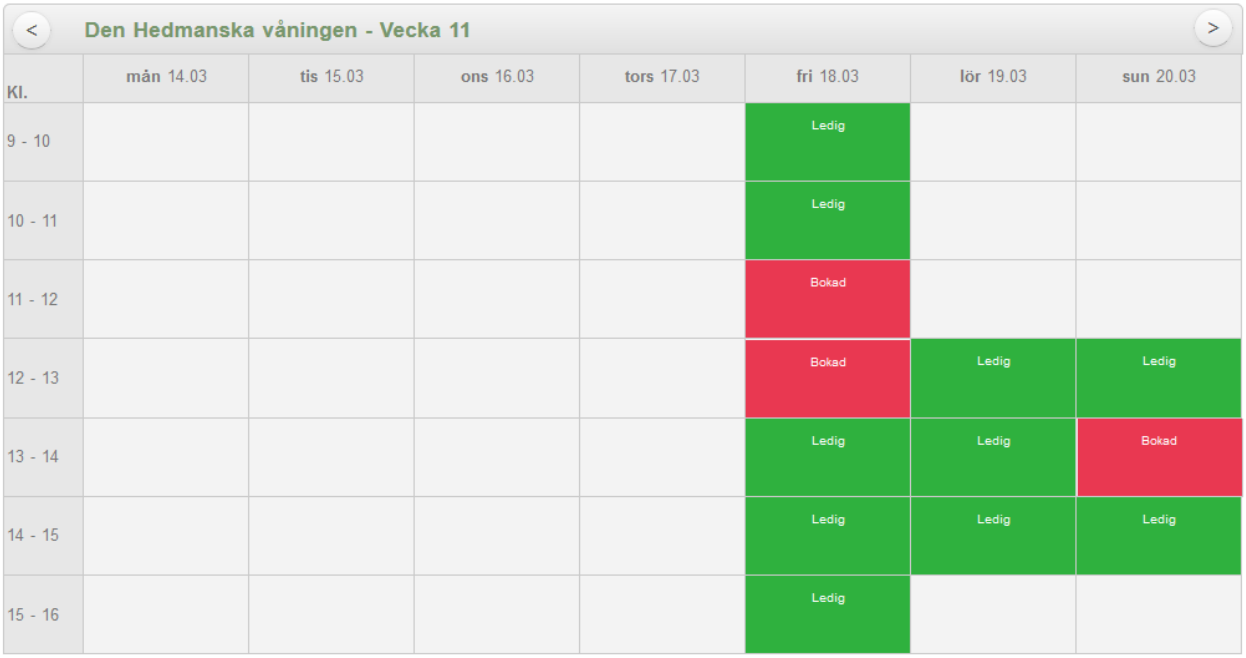
Beroende på vilken av de tre språkfilerna som inkluderats visas texten på det språket. Med detta system behövs endast en version av sidan som sedan kan visas på alla språk.

Motsvarande system finns även i databasen där de editerbara texterna finns sparade i kolumner enligt formatet namnse, namnfi och namneng. Beroende på vilket språk som valts hämtas texten från korrekt kolumn.

3.2.4 Kalender

Kalendern har sju kolumner för veckodagar med start från måndag och sju rader som visar tider från 09.00 till 16.00 med en timmes mellanrum, dvs. 09.00 - 10.00, 10.00 - 11.00 osv. De bokningsbara tidpunkterna är från tisdag till fredag 09.00 - 16.00 och på lördag och söndag är tiderna 11.00 -15.00. Måndagar är inte bokningsbara. Vidare så går det endast att boka tider fr.o.m. tre dagar framåt, dvs. på en tisdag är endast lördag och framåt bokningsbara.

De tidpunkter som går att boka indikeras med gröna rutor och texten Ledig och de som inte går att boka har röda rutor och texten Bokad. Då användare markerar en tidpunkt så ändras färgen till gult för att indikera att rutan är vald. Användare kan även markera flera tidpunkter på olika dagar. Det finns däremot ingen funktion för att välja olika grupper för de olika tiderna, utan alla bokningar hamnar under samma grupp. En bokning kan inte heller göras för olika veckor samtidigt.



Figur 8 kalendern

Då användaren har gjort sitt val, fyllt i formuläret och skickat den, visas en dialogruta med information om att bekräftelse har skickats som e-brev till den givna adressen. Då användaren klickar på den länk som finns i bekräftelsen navigeras webbläsaren till den vecka och utställning som användaren valt och den tidpunkten visas nu som röd för att indikera att den blivit bokad. En ny dialogruta visas för att informera användaren om att bokningen nu är bekräftad. Ett nytt e-brev blir då skickat med uppgifter om bokningen.

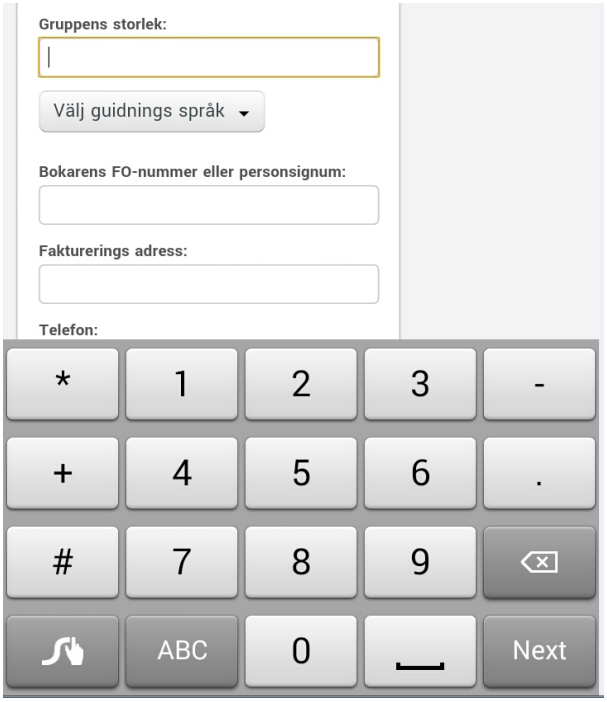
Samma information som visas i dialogrutorna finns även på sidan ifall användarens webbläsare blockerar dialogrutorna. Dialogrutorna samt texten i e-brevet är på det språk som användare valt. Ifall inget språk valts används finska som standard.

3.2.5 HTML5 och kontaktuppgifter

Webbsidan har ett formulär där besökare fyller i sina kontaktuppgifter samt information om bl.a. storleken på gruppen och guidningsspråk.
HTML5 introducerade funktionen *required* för formulär som beskrivs närmare med kodexempel i nästa kapitel. Med den kan man välja vilka fält som inte kan lämnas tomma. Ifall ett fält satt som *required* lämnas tomt och användaren försöker skicka formuläret, visas en uppmaning att fylla i det tomma fältet.

HTML5 utökade även *input* med nya attribut för formulärer som går att kombinera med *required* för mer specifik validering. *Requiered* själv kräver endast att ett *input* fält inte saknar innehåll för att vara giltig. Exempel på nya attribut som sidans formulär använder är *tel*, *email* och *number*. Andra vanliga attribut är t.ex. *url*, *date*, *time* och *range*. (MacDonald, 2011, s. 123)

Tidigare användes endast JavaScript för att validera formulärer på klientsidan. Med HTML5:s nya attribut kombinerat med *required* kan man enkelt få mer omfattande valideringsregler för ett formulärs fält. Av de attribut som formuläret använder har däremot *tel* inga specifika regler då dessa varierar från land till land och kan innehålla t.ex. + tecken. Fördelen med *tel* är dock att moderna webbläsare känner igen den och vissa Android modeller, samt iPhone, byter ut tangentbordet till en numerisk layout för *tel*, vilket underlättar när man ska skriva in ett telefonnummer. (MacDonald, 2011, s. 124)



Figur 9 Androids numeriska tangentbordslayout

Number tillåter endast numerisk inmatning och, beroende på modell och webbläsare, ändras tangentbordet på samma sätt som med *tel*. Vidare lägger vissa webbläsare även till en *spinbox* till *number* för att underlätta inmatning. Andra valideringsregler som kan kombineras med *number* är *min* och *max*. Med dessa kan man specificera de högsta och lägsta tillåtna värdena. I formuläret används *min* regeln endast för att grupper med en eller flera personer ska gå att registrera. (MacDonald, 2011, s. 126)

Email erbjuder en enkel valideringsregel för den givna adressen. Den kontrollerar inte att den är en riktig adress, däremot kontrollerar den att formatet är korrekt vilket underlättar för användaren att undvika t.ex. mellanslag eller motsvarande i adressen. Endast inmatning i formatet namn@exempel.com är giltiga. Som med *tel* och *number* erbjuder vissa telefoner, beroende på modell, en skild tangentbordslayout för *email* fält. (MacDonald, 2011, s. 125)

Andra fördelar med de nya attributen är att vissa webbläsare kan, ifall informationen finns sparad och den känner igen de olika *input* fälten, fylla i formulären automatiskt. (Keith, 2010, s. 44)

De nya attributen erbjuder en smidig lösning både för den som gjort formuläret och för att meddela användaren om de värden som skrivits in är i korrekt format samt ifall någon av dem lämnats tom eller skrivits i fel format.
Däremot kan den inte helt ersätta den mer traditionella valideringen med JavaScript. Ifall en användare fyller i formuläret med en webbläsare som inte stöder de nya attributen så skulle ingen validering ske. På grund av det har sidan även en JavaScript-funktion som kontrollerar att alla fälten är ifyllda och meddelar användaren ifall någon lämnats tom.

Sidan har även ett dolt *input* fält som endast kan fyllas i genom att användaren har markerat en eller flera tider från kalendern. Detta ger dels användaren en påminnelse att ingen tid har valts samt gör det svårare för enklare spambottar att fylla i formuläret. Detta är en variant av ett enkelt Turing test som vissa sidor använder. Testet går ut på att ha ett dolt fält som en mänsklig användare inte ser och därmed inte kommer att fylla i. En sambot däremot ser även dolda fält och kan inte veta vilka som måste vara ifyllda, så den fyller då i alla existerande fält. Sidan kör sedan en kontroll för att se ifall det gömda fältet är ifyllt, då är det högst sannolikt en maskin som fyllt i formuläret och den kan därmed kasseras.
Sidans version är lite annorlunda då fältet måste vara ifyllt för att vara giltigt, men kan endast fyllas i genom att användaren markerar en bokningsbar tid på kalendern. (Psinas, 2012)

HTML5 kombinerat med JavaScript räcker dock inte till för att göra formuläret säkert. Alla värdena måste till sist kontrolleras före dom skrivs in i databasen för att minimera risken för bl.a. *SQL injection*. *SQL injection* är den typ av attack där okontrollerade värden skickas in till databasen. (MacDonald, 2011, s. 113)

Ett typiskt exempel är när en användare loggar in till en sida och skriver in *' or '1'='1* som användarnamn och lösenord som i denna SQL *query* exempel.

Kodexempel 10

```
SELECT * FROM USERS WHERE USER='' AND PASS='';
```

Original.

```
SELECT * FROM USERS WHERE USER='' or '1'='1' AND PASS='' or '1'='1';
```

Modifierad.

Original kontrollerar ifall de givna uppgifterna finns i databasen. Modifierad kontrollerar användarnamn och lösenord, men då användare skrivit in *' or '1'='1* i båda fälten resulterar det i en *query* där 1 alltid är ett korrekt användarnamn och lösenord.

För att undvika att SQL-kod som kan köras blir skickat med till databasen från formuläret används PHP-funktionen *mysql_real_escape_string()*. Funktionen ser till att specialtecken inte kan tolkas som SQL-kod. Den bör inte användas fr.o.m. version 5.5.0 av PHP och funktionen har tagits bort fr.o.m. version 7.0.0. MySQLi och PDO_MySQL har funktioner som ersätter *mysql_real_escape_string()*, men den server sidan ligger på stöder dock inte dessa alternativ som bör användas. (Clark, 2012, s. 511)

3.3 Kodexempel

Här följer exempel på de funktioner som nämndes tidigare, med ett stycke kod följt av en förklaring av vad den gör.

3.3.1 Form

Form är HTML-elementet som innehåller alla *input* fälten. Elementet har även information om bl.a. vad som skall ske när användare skickar formuläret.

Kodexempel 11

```
<form action="index.php" enctype="multipart/form-data" method="post"
onsubmit="return validateform();">
<ul>
<label for="l-phone"><?php echo $langarray[19]; ?></label>
<input type="tel" name="phone" id="l-phone" required>
</ul>
</form>
```

I exemplet från sidan säger *method=post* att den skall skicka vidare all information från formuläret. *Action* attributet säger vilken fil som skall ta emot och hantera den information som skickas från formuläret. I exemplet är index.php, vilket är samma fil som själva formuläret finns i, satt att ta emot informationen. Detta sker via PHP-funktionen *$_POST* som beskrivs närmare längre ner.

Ul står för oordnad lista och används med *list item* <*li*></*li*> HTML-elementen för att organisera saker i listor. *Label* används för att ge ett fält en rubrik och kan med HTML5 associeras med ett *input* element ifall *label* elementets *for* tag och *input* elementets *id* tag har samma värde. Detta underlättar t.ex. för skärmläsare att förstå vilka element som hör ihop. Själva texten för *label* elementet hämtas från språkräckan.

Type säger vilken typ av fält det är och webbläsare som understöder dessa kan anpassa hur användare fyller i dem. *Required*, som kan skrivas som required, required="" eller requierd="required", säger att fältet inte kan lämnas tomt.

Här är ett exempel från den kod som skriver informationen från formuläret till databasen.

Kodexempel 12

```
if(isset($_POST['name']))
        $name = mysql_real_escape_string($_POST['name']);
        $gname = mysql_real_escape_string($_POST['gname']);
        $gsize = mysql_real_escape_string($_POST['gsize']);
        $glang = mysql_real_escape_string($_POST['glang']);

        mysql_query("INSERT INTO tblUsers(name,gname,gsize,glang)
        VALUES('".$name."','".$gname."','".$gsize."','".$glang."')") OR die (
        mysql_error());
        header("Location: index.php?museum=".$museumId."&exhibition=".$revyid."
        &week_no=".$splittedDate[2].$checkMail);
}
```

Exemplet börjar med att kontrollera med *isset()* funktionen ifall ett formulär har blivit skickat. Om så är fallet så tar den emot alla *$_POST* värden som skickats. *Mysql_real_escape_string* körs som kontroll på alla värdena för att de skall vara säkra innan de sparas i sin egen variabel.

Dessa skickas sedan till databasen med *mysql_query insert into* funktionen. Ifall det finns syntax fel i koden visar *mysql_error()* var dessa finns. Till sist körs *header location* funktionen som navigerar användaren vidare ifall all kod körts utan några problem.

Form elementet är även satt att köra JavaScript-funktionen *validateform()* när användare skickar den. Som exempel kontrolleras fältet *fname* med följande kod.

Kodexempel 13

```
if(document.bookingform.elements['fname'].value=="")
	{
		dlg2.show();
		return false;
	}
```

Koden kontrollerar ifall ett formulär med namnet *bookingform* har ett fält som heter *fname* och att värdet i fältet inte är “”, alltså tomt. Om fältet är tomt körs funktion, *dlg2.show()*, annars gör den inget.

Funktionen *dlg2.show()* kör följande kod.

Kodexempel 14

```
<div id="dialogBokning">
	<div class="hd">Fyll i alla uppgiter</div>
	<div class="bd">
	Fyll i email.
	</div>
</div>
<script type="text/javascript">
var dlg2 = new YAHOO.widget.Dialog("dialogBokning", {
		width:"525px",
		fixedcenter:true,
		modal:true,
		visible:false,
		zindex:20
	});
	dlg2.cfg.queueProperty("buttons", myButtons);
	dlg2.render();
</script>
```

Först är HTML-elementet av typen *div* definierat med namnet *dialogBokning* som innehåller den text som visas åt användare. Koden under säger att då *input* fältet i formuläret lämnats tomt så skall en dialogruta med en specificerad position samt storlek öppnas och dess innehåll skall vara *div* elementet *dialogBokning*.

4 Admingränssnitt

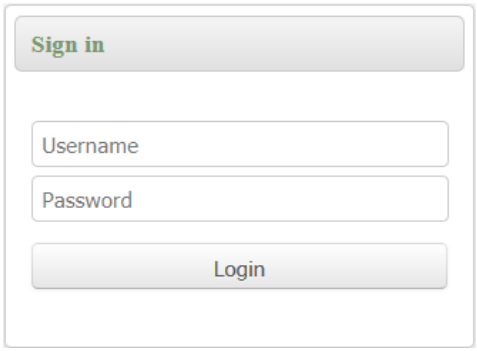
Webbsidan har ett admingränssnitt som man kommer åt genom att logga in på sidan. När användare loggar in skapas en *session cookie* som säger att användaren har rättigheter som administrator. Sidan har vissa funktioner och grafiska delar som endast visas eller aktiveras då användaren är inloggad.

4.1 Syfte

Syftet med administratörsgränssnittet är att ge de anställda vid museet möjlighet att själv kunna uppdatera samt lägga in nytt material på webbsidan. Applikationen har även information och funktioner som finns tillgängligt endast för administratörer. Exempelvis visas information om vem som gjort bokningar och ifall en administratör gör en bokning behövs ingen bekräftelse.

4.2 Funktioner

För att komma åt administratörgränssnittet loggar användaren in till webbsidan. De funktioner som finns tillgängliga för administratörer blir då aktiverade. De delar av applikationen som har utökad funktionalitet är kalendern samt formuläret för kontaktuppgifter. Som administratör har användaren även tillgång till en kontrollpanel, statistiksidan samt bokningslistan.



Figur 10 Inlogningsruta

4.2.1 Kontrollpanelen

Då användaren är inloggad finns kontrollpanelen tillgänglig på sidans högra kant eller under kalendern.

Figur 11 Kontrollpanel

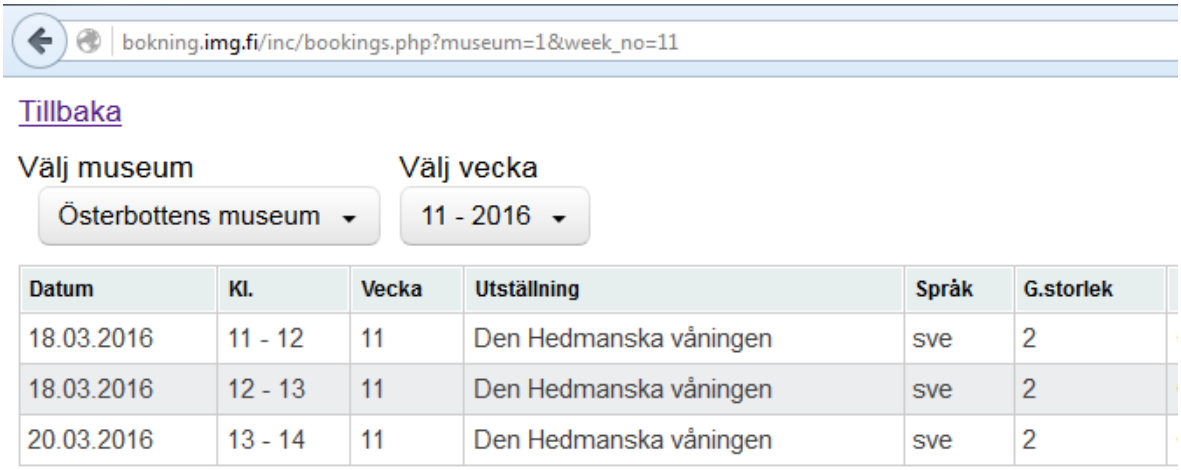
Panelen innehåller länkar till statistiksidan och bokningslistan samt funktioner för att editera eller ta bort utställningar och att lägga till nya utställningar. För att editera en existerande utställning väljs den från kalenderns filtreringsalternativ.

4.2.2 Kalendern

Kalendern har en funktion för att ta bort bokningar genom att dubbelklicka på en bokad ruta. En dialogruta visas då där administratören ombeds bekräfta borttagningen. Administratören kan även markera en bokad tidpunkt för att visa kontaktuppgifter för den som gjort bokningen. Dessa listas då ovan kontaktformuläret.

4.2.3 Bokningslista

Webbsidan har en skild funktion för att lista alla gjorda bokningar. Via kontrollpanelen kan administratören navigera vidare till bokningslistan. Listan hämtas från databasen och ifall administratören har valt ett museum filtreras listan enligt det valda museet. Nuvarande vecka väljs som standard. Administratören kan välja att filtrera listan enligt museum och månad.



Datum	Kl.	Vecka	Utställning	Språk	G.storlek
18.03.2016	11 - 12	11	Den Hedmanska våningen	sve	2
18.03.2016	12 - 13	11	Den Hedmanska våningen	sve	2
20.03.2016	13 - 14	11	Den Hedmanska våningen	sve	2

Figur 12 bokningslista

Listan innehåller all den information som skickats från formulären samt de utställningar och museer bokningarna gjorts till. Österbottens museum har en skild kolumn för utställningar då de andra museerna kan vara länkade endast till en utställning.

4.2.4 Kod

Detta kodexempel visar hur språkalternativen sätts.

Kodexempel 15

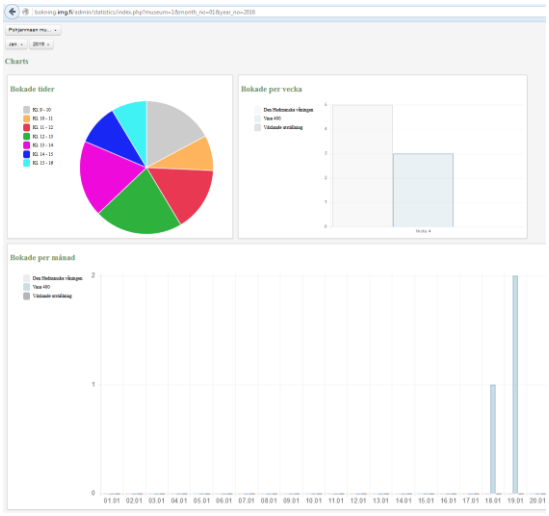
```
$defaultlang = "fi";
if(empty($_SESSION['lang'])) {
    //from cookie
    if($_COOKIE['lang']=='fi' || $_COOKIE['lang']=='se' || $_COOKIE['lang']=='eng') {
        $_SESSION['lang']=$_COOKIE['lang'];
    }
    else {
        $inTwoMonths = 60 * 60 * 24 * 60 + time();
        setcookie("lang", $defaultlang,$inTwoMonths);
        $_SESSION['lang']=$defaultlang;
    }
}



//setlang
if(isset($_GET['setlang'])) {
    if($_GET['setlang']!='fi' && $_GET['setlang']!='se' && $_GET['setlang']!='eng')
        $_GET['setlang']=$defaultlang;


    $inTwoMonths = 60 * 60 * 24 * 60 + time();
    setcookie("lang", $_GET['setlang'],$inTwoMonths);
    $_SESSION['lang']=$_GET['setlang'];
}
```

I PHP-variabeln *$defaultlang* är värdet satt till *fi*, vilket betyder att ifall inget språk valts väljs finska som standard språk. Då användaren besöker sidan skapas en *session* med information om bl.a. språkvalet som sparas i *$_SESSION['lang']*. Ifall *$_SESSION['lang']* inte har ett värde kontrolleras ifall det finns en *$_COOKIE['lang']* från tidigare besök. Om en sådan finns sparas språkvalet från denna i *$_SESSION['lang']* variabeln och om ingen sådan finns får den standardspråket som sitt värde. Sist kontrolleras ifall variabeln *$_GET['setlang']* har ett värde, om så är fallet har användaren valt eller ändrat språk. Variabeln kontrolleras så att

värdet är ett giltigt och om inte väljs finska. Om värdet är giltigt sätts detta språk för sessionen och webbsidorna visas på detta språk.

5 Statistik

Statistiksidan använder verktyget *Chart.js* som ger designare och utvecklare utan djupare kunskap om programmering möjlighet att integrera professionella grafer i sina projekt.



Figur 13 Statistik sidan

5.1 Syfte

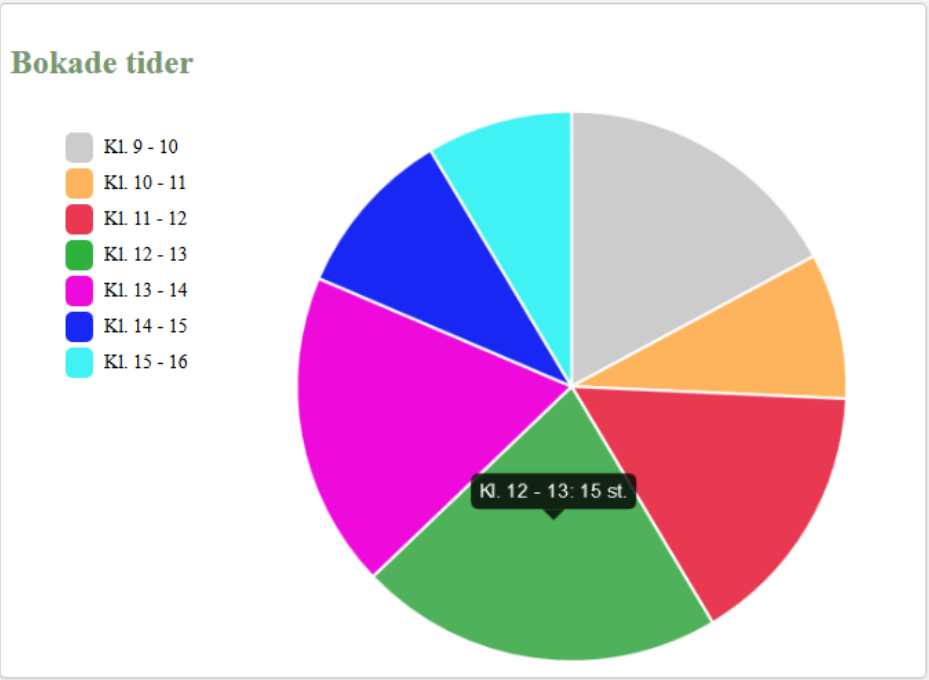
Sidan ger möjlighet att se statistik över gjorda bokningar och använder *Chart.js* verktygets stapel och cirkeldiagram för att visualisera data.

5.2 Funktioner

Sidan visar tre olika diagram samt de tre filtreringsalternativen där en användare inloggad som administratör kan välja mellan museum, år och månad. Listan med museer hämtas från den statiska räckan och åren från de år som i databasen som innehåller bokningar.

5.2.1 Cirkeldiagram

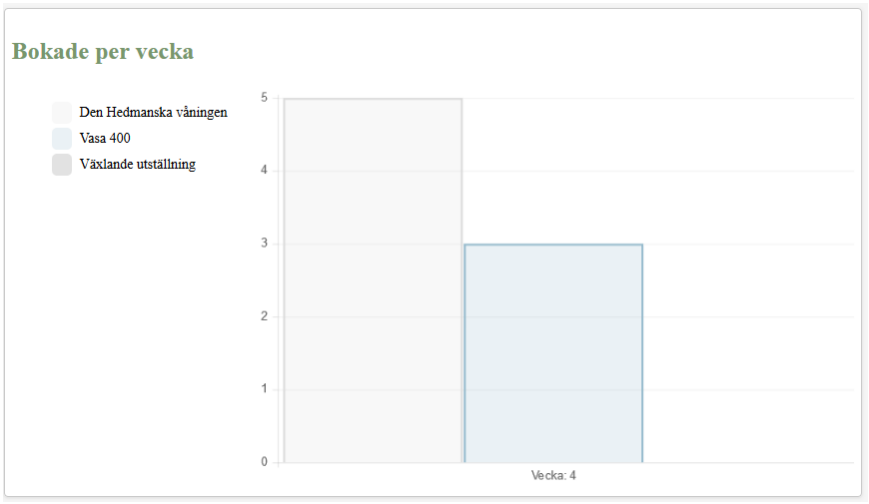
Diagrammet är uppdelat enligt vilka tider som blivit bokade. I exemplet har 15 st. bokningar gjorts för tiden 12.00 - 13.00. Diagrammet går att filtrera enligt år och museum.



Figur 14 cirkeldiagram

5.2.2 Stapeldiagram för veckor

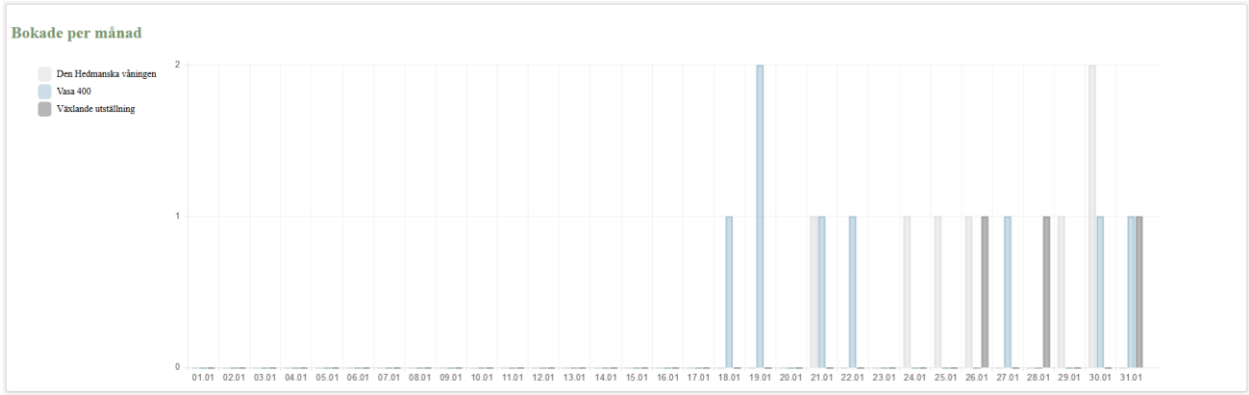
Den andra rutan innehåller ett stapeldiagram över gjorda bokningar. Staplarna visas veckovis och kan filtreras enligt museum, månad och år. För Österbottens museum visas skilda staplar för de olika utställningarna, för de andra museerna så visas endast en stapel.



Figur 15 Stapeldiagram

5.2.3 Stapeldiagram för månader

Den sista rutan visar månadsvis ett stapeldiagram över gjorda bokningar. Diagrammet går att filtrera enligt museum, månad och år. Diagrammet visar hela månader och har värdet noll för de dagar då inga bokningar gjorts. Som med det föregående diagrammet har Österbottens museum skilda kolumner för sina olika utställningar.



Figur 16 Stapldiagram

5.3 Kodexempel

Statistiksidan använder SQL *querys* som hämtar data till diagrammen. Dessa använder JavaScript till konfigurationen så all data som hämtas från databasen måste konverteras till ett hanterbart format via PHP-funktioner.

5.3.1 SQL

Statistik sidan använder ett antal SQL *queries* för att hämta den data som diagrammen använder.

Kodexempel 16

```
$resultMonth = mysql_query("SELECT week, DATE_FORMAT( date, '%d' ) as date, exhebitionId,
tblBookings.museumId, COUNT( * ) AS bookings, tblUtstallning.id,
tblUtstallning.descriptionSe
FROM tblBookings
LEFT JOIN tblUtstallning ON tblUtstallning.id = tblBookings.exhebitionId
WHERE date LIKE '".$_GET['year_no']."-".$_GET['month_no']."-%' AND tblBookings.museumId =
'".$_GET['museum']."'
GROUP BY date");
```

Exemplet hämtar den data som visas i stapeldiagrammet för månader och sparar den i PHP-variabeln *$resultMonth*. *Select* säger att från tabellen *tblBookings* skal kolumnerna vecka, datum, utställningens id nr, museets id nr och antalet bokningar hämtas. En utställnings id hämtas även från tabellen *tblUstallning* tillsammans med namnet på utställningen.

Funktionen *date_format()* säger i vilket format datum skall hämtas. Som standard sparas alla datum i databasen enligt formatet yyyy.mm.dd, som exempel 2016.02.13. Diagrammet behöver dock endast “dd” delen, dvs. den dag bokningen är gjord på. Med *date_format()* kan man specificera att endast dag delen av ett datumet skall hämtas.

Count() funktionen räknar förekomsten av specifika värden. I exemplet grupperas resultatet enligt datum, där funktionen räknar förekomsten av ett visst datum. Ifall exempelvis 2016.02.13 förekommer tre gånger betyder det att tre bokningar gjorts på den dagen och *count()* returnerar då värdet 3.

Tabellen *tblBookings* innehåller inga namn på utställningarna utan endast deras id. Med *left join* funktionen hämtas dessa från tabellen *tblUtstallning* och kopplas sedan ihop via *id* kolumnerna.

Where delen får sina värden via PHP *$_GET* funktionen från alternativen i rullgardinsmenyn. Resultatet filtreras därmed enligt museum, år och månad beroende på vad användaren valt.

5.3.2 PHP

Ett exempel på hur resultat konverteras till hanterbar data.

Kodexempel 17

```
function padDates($dataArray,$numberOfDays) {
	$array = array();

	for ($i = 0; $i <= 31; $i++)  {
	$array[]=array('date'=>$dataArray[$i]['date'],'dataset'=>$dataArray[$i]['bookings'])
;
	}

	$dates = array();
	for($i = 1; $i <= $numberOfDays; $i++){
		$dates[$i]=0;
	}

	for($i = 0; $i <= count($array); $i++){
		$dateSplit = $array[$i]['date'];
		for($g = 0; $g <= $numberOfDays; $g++){
			str_pad($i, 2, '0', STR_PAD_LEFT);

			if($g == $dateSplit)
			{
				$dates[$g]=$dataArray[$i]['bookings'];
			}
		}
	}
	return $dates;
}
```

Diagrammet som visar bokningar månadsvis skall ha värdet noll för alla de dagar då inga bokningar skett. För att få den data som hämtas från databasen, som bara innehåller de dagar då en eller flera bokningar skett, i korrekt format körs funktionen *padDates()*. Funktionen tar som in parametrar en räcka med de datum som bokningar gjorts samt antalet bokningar.

Kodexempel 18

```
$dataArray = array();
while($dataArray[] = mysql_fetch_array($resultMonth));
```

En *while loop* körs för att spara värdena från databasen till räckan *$dataArray* som funktionen kan hantera. Den andra in parametern *$numberOfDays* använder PHP-funktionen *date()* för att få ut antalet dagar för den valda månaden. En *for loop* körs sedan i funktionen *padDates()* för att skapa en tvådimensionell räcka *$array* som för varje position innehåller värdena datum och antalet bokningar. En annan räcka, *$dates*, skapas och en loop körs som ser till att dess storlek motsvarar antalet dagar i den valda månaden samt instansierar alla positioner med värdet 0. De två räckorna kombineras till sist i en loop där varje position i *$array* som har samma värde som positionen i *$dates* sparas antalet bokningar. Annars är antalet bokningar noll. Funktionen returnerar *$dates* som nu innehåller en lista av storleken *$numberOfDays* samt antalet bokningar för de dagar som har en bokning.

Exempel på de tre räckorna.

Tabell 3 och 4 exempel på *$array* och *$numberOfDays* som *padDates()* skapar

Dag (date)	Antal bokningar (dataset)
2	6
4	1
9	5
12	3
...	...

Day	Bookings
1	0
2	0
3	0
4	0
...	...

De två räckorna kombineras sedan till *$dates*

Tabell 5 kombinerad tabell

Datum	Antal bokningar
1	0
2	6
3	0
4	1
5	0
6	0
7	0
8	0
9	5
10	0
11	0
12	3
..	..

Variabeln *$dates* som returneras innehåller dock endast kolumnen antal bokningar, men kolumnen datum finns med i exemplet för att illustrera relationen mellan de två tidigare tabellerna.

5.3.3 JavaScript

Räckan som returneras från *padDates()* sparas sedan i *Chart.js* variabler för att sedan visas på sidan. Ett exempel på Chart.js diagram ser ut som följande.

Kodexempel 19

```
var monthlyData = {
    labels: [<?php foreach($numberOfDays as $a){ echo "\"",str_pad($a, 2, '0',
STR_PAD_LEFT),".".$_GET['month_no']."\",";} ?>],
    datasets: [
        {
            label: "<?php echo $museums[$_GET['museum']]; ?>",
            fillColor: "rgba(220,220,220,0.5)",
            strokeColor: "rgba(220,220,220,0.8)",
            highlightFill: "rgba(220,220,220,0.75)",
            highlightStroke: "rgba(220,220,220,1)",
            data: [<?php foreach($dates as $a){ echo "\"",$a,"\",";} ?>]
        }
    ]
};
```

Variabeln *monthlyData* har en *label* som representerar x-axeln på diagrammet och får sina värden från PHP-variabel *$numberOfDays*. En *foreach* loop körs som listar alla månadens dagar i formatet “1,2,3,...”. Funktionen *str_pad()* lägger till nollor för 1 - 9 så att de visas som 01, 02, 03 etc. Efter varje dag listas den valda månaden. Resultatet blir datum i formatet 01.02, 02.02, 03.02, ..., 10.02 etc.

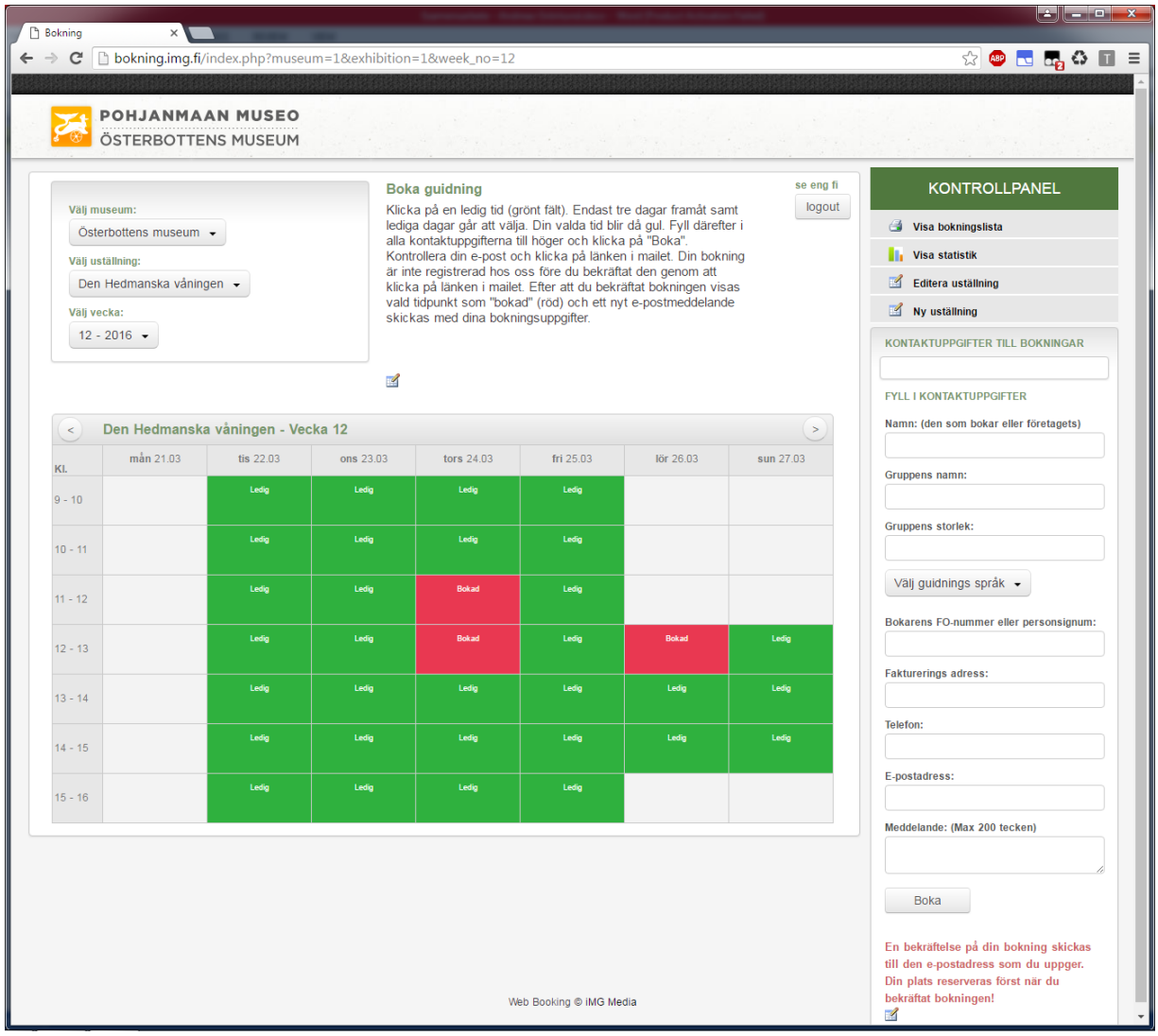
Datasets innehåller värden som visas i diagrammet. Konfigurationsattributet *Label* är namnet på kolumnen och hämtas med *$_GET* från det valda museet. De fyra följande attributen säger vilken färg kolumnerna skall ha och det sista attributet *data* itererar genom $*dates* räckan som skapades i föregående exempel. Då denna har samma storlek som antalet dagar i den valda månaden som listas i x-axeln kombineras alla bokningar automatiskt enligt korrekt dag.

6 Diskussion

Webbapplikationen gjordes på en botten av ett tidigare projekt. Detta gjordes föra att spara tid under utvecklingen. Det visade sig dock att det mesta av den ursprungliga koden kom att skrivas om och det möjligen skulle ha gått snabbare att göra examensarbetet som ett helt nytt projekt. Det som dock sparade tid var att använda sig av färdiga funktioner som *Chart.js* diagramverktyg och *SelectBoxIt* som används till rullgardinsmenyerna för att göra dessa mobilanpassade.

6.1 Resultat

Resultatet av examensarbetet blev en webbapplikation som innehåller all den funktionalitet som behövs för att ge användare möjlighet att via Österbottens museums webbsidor boka guidade rundturer. Vidare ger den möjlighet att administrera utställningar och bokningar. Applikationen visar även statistik över bl.a. antalet bokningar. Då detta skrivs är applikationen ännu inte i användning utan håller på att testas. Efter att den testats och ifall inga nya funktioner skall läggas till eller existerande modifieras är applikationen redo att tas i bruk.



Figur 17 webbapplikationen med administratorgränsnit

Statistiksidan krävde komplicerade SQL *queries* för att samla och formatera all den data som diagrammen behövde, så jag lärde mig mycket om dessa. Kalendern använder datum till många av sina funktioner samt för filtreringen. För att få dessa korrekt användes många av PHP:s egna funktioner för datum som krävde en del tid att lära sig. Vidare lärde jag mig hur man enkelt kan utöka funktionaliteten för en webbapplikation med färdiga verktyg, som t.ex. *Chart.js.*

6.2 Vidareutveckling

Både kalendern och statistiksidan har goda möjligheter att utvecklas vidare. Kalendern har bl.a. en *drag and drop* funktion som för tillfället inte är i användning. Den kunde t.ex. användas för att ge en administratör möjlighet att flytta en bokning från en tidpunkt till en annan enbart genom att flytta rutan.

Kontaktuppgifter för en bokning går för tillfället inte att editera, utan en ny bokning måste i så fall göras och den gamla tas bort. En lösning till det skulle vara att ge kontaktformuläret funktionen att editera dessa. Administratören kunde t.ex. markera en bokning och formuläret skulle automatiskt ladda uppgifterna från bokningen. Dessa kunde då editeras och sparas utan att en ny bokning behövde göras.

Statistik sidan har många olika möjligheter att utvecklas. För tillfället finns där endast tre diagram. Dessa kunde utökas till att visa bl.a. vilka gruppstorlekar som är de vanligaste eller vilka dagar de flesta bokningar sker. Även filtreringen kunde utökas med flera alternativ. Administratören kan i nuläget filtrera informationen enligt museum, månad och år. Andra möjligheter kunde vara t.ex. dag, språk och tid.

För tillfället finns information endast om existerande utställningar. Det blev aldrig specificerat hur denna del skulle fungera, men som ett alternativt skulle möjlighet att inkludera information om äldre utställningar kunna vara intressant. Detta skulle dock endast gälla de museer som har växlande utställningar.

Andra tillägg till sidan skulle vara att lista statistiken i tabellformat. Detta skulle underlätta för administratören att använda informationen till annat. Denna funktion kunde sedan vidareutvecklas till att ge administratören möjlighet att exportera statistik om bokningar i olika format.

Under utvecklingen kom möjligheten att samtidigt boka flera tider men på olika veckor upp. För tillfället går det endast att göra bokningar för en vecka och ifall användaren vill lägga till ytterligare bokningar för andra veckor måste en ny bokning göras för varje vecka. Förutom att göra bokningar för olika veckor kom även möjligheten att specificera olika grupper upp. Ifall användaren väljer flera olika tidpunkter blir dessa nu bokade under samma grupp.

7 Källförteckning

MacDonald, M., 2011. *HTML5: The Missing Manual*. (2. uppl.) California: O'Reilly Media, Inc

Keith, J. 2010. *HTML5 for Web Designers*. (1. uppl.) New York: A Book Apart LLC

Psinas, M., 2012. *Easy Spam Prevention Using Hidden Form Fields*. [Online] http://www.sitepoint.com/easy-spam-prevention-using-hidden-form-fields [hämtat: 10.3.2016].

Clarke, J., 2012. *SQL Injection Attacks and Defense* (2. uppl.) Boston: Syngress

ISO week date [Online] https://en.wikipedia.org/wiki/ISO_week_date#First_week [hämtat: 20.12.2015].

Beighley, L. & Morrison, M., 2014. *Head First PHP & MySQL*. (1. uppl.) Sebastopol: Oreilly & Associates Incorporated

Haverbeke, M., 2014. *Eloquent JavaScript: A Modern Introduction to Programming*. (2. uppl.) San Francisco: No Starch Press

Conformance: requirements and recommendations [Online] https://www.w3.org/TR/REC-html40-971218/conform.html [hämtat: 15.3.2016].

HTML5 Introduction [Online] http://www.w3schools.com/html/html5_intro.asp [hämtat: 15.3.2016].

CSS Introduction [Online] http://www.w3schools.com/css/css_intro.asp [hämtat: 15.3.2016].